CSCI368 Network Security
Session 2 2019

Assignment 1 (20 Marks)

Submission Due: 28 April 2019 23:55 (SG Time)

Objectives

On completion of this assignment you should be able to:

- Understand some basic techniques for building a secure channel.
- Understand network programming.

Write (Java or C/C++) UDP programs allowing two parties to establish a secure communication channel. For simplicity, let us call the programs "Host" and "Client", which are executed by Alice and Bob, respectively.

Alice and Bob share a common password PW, which contains 6 numeric characters, and the parameters (p, g) for Diffie-Hellman key exchange. They want to establish a secure communication channel that can provide data confidentiality and integrity. They aim to achieve this goal via the following steps: (1) use the shared information to establish a shared session key; (2) use the shared session key to secure the communication.

Step 1 is done via the following key exchange protocol:

1: $A \rightarrow B: E_{PW}(g^x \bmod p)$

2: $B \rightarrow A: E_{PW}(g^y \bmod p)$

Alice and Bob then compute the shared key as $K = H(g^{xy} \bmod p)$ where H denotes a cryptographic hash function. Alice and Bob decide to use RC4 as the encryption function, and SHA-1 as the Hash function.

After establishing the session key, **step 2** is achieved as follows:

1. whenever Alice wants to send a message M to Bob, Alice first computes H = Hash(K||M), and then computes $C = E_K(M||H)$ and sends C to Bob. Here || denotes the string concatenation.
2. upon receiving a ciphertext C, Bob first runs the decryption algorithm to obtain $M||H = D_K(C)$. After that, Bob computes H' = Hash(K||M) and checks if H = H'. If the equation holds, then Bob accepts M; otherwise, Bob rejects the ciphertext.
3. the same operations are performed when Bob sends a message to Alice.

Implementation guidelines

- Place Host and Client in two separate directories: Alice and Bob. The shared information (PW, p, g) is put in a text file under each directory.

- Alice executes Host.

  - Host is running and listening to the opened port (you need to select a port for your code).

- Bob executes Client.

- Client (Bob) sends a connection request to Host.

- Client is ready and listens to the port.

- Host generates a random x, computes g^x mod p, encrypts it using RC4, and sends the ciphertext to Client.

- Upon receiving the message from the Host, Client decrypts the message to obtain g^x mod p. Client also generates and sends the encryption of g^y mod p to Host. Client then computes the shared key K.

- Upon receiving the message from Client, Host performs the decryption to obtain g^y mod p and then computes the shared key K.

- Now, the secure channel is established.

  - Either Alice or Bob can send a message encrypted and authenticated by the key K. They type the message on their own terminal. The message is processed by their code (Host or Client) according to the step 2 given above.

  - The received message is printed on the screen if decryption is successful. Otherwise, print "decryption error" on the screen.

  - To quit the program, the client should type "exit".

Coding requirement:

You need to write three programs: Gen, Host and Client. The system parameter generation function Gen should generate the common parameters (p, g) for the Diffie-Hellman key exchange. The Diffie-Hellman modulus p must have at least 32 bits and g is a generator of the multiplicative group Z*p.

You can choose to use some existing libraries or free source code to implement RC4 and SHA-1. You should cite the source if you use a downloaded code.

How to run?

Your programs should run according to the protocol. Host and Client should be executed on different windows. For convenience of marking, please use the local IP: 127.0.0.1 for the submitted version. For simplicity, there is no GUI required in this assignment. That is, messages are simply typed on the window and printed on the receiver's window. The looping should continue until the moment the user types "exit" to exit.

Files to be submitted:

All source codes.

A readme file (text/ACSII only): instructions about how to compile and run your code.

A Makefile: for C++ programmers. Alternatively, you provide the compilation instruction in the readme.

Submission

Compress all the files to be submitted into a zip file and submit it via the submission link provided in the Moodle site.

Late Submission: Penalty is 25% deduction per day (including weekends).

Marking

Mark distribution:

1. Diffie-Hellman parameter generation: 4 marks
2. Key Establishment (Step 1): 8 marks
3. Data Encryption/Decryption (Step 2): 8 marks

The code that cannot be compiled or executed will receive a zero mark.

Plagiarism

A plagiarised assignment will receive a zero mark and be penalised according to the university rules. Plagiarism detection software may be used.